GIM Tool: A Global Icosahedral Atmospheric Model Viewer

Authors: Evan Polster, Jeff S Smith, Ning Wang
CIRA researchers at Global Systems Division (GSD) of Earth System Research Laboratory (ESRL)

GSD is developing two global icosahedral weather models: the finite-volume flow-following icosahedral model (FIM) [1] and the non-hydrostatic icosahedral model (NIM). Our group was tasked with developing an innovative 3D viewing application for the purpose of displaying such global model data, which would be web-based, and intuitive to use as an outreach and diagnostic tool.

What Is An Icosahedral Grid?

The icosahedral grid [2] is created by recursively bisecting the 20 triangular faces of the original regular polyhedral (icosahedron) and projecting bisection points to the sphere. The number of recursive refinements is referred to as grid level. The number of grid points is defined by $N = 10 * 2^{2g} + 2$, where g is the grid level. The areas of grid points are the spherical Voronoi cells defined by the grid points (Figure 1).

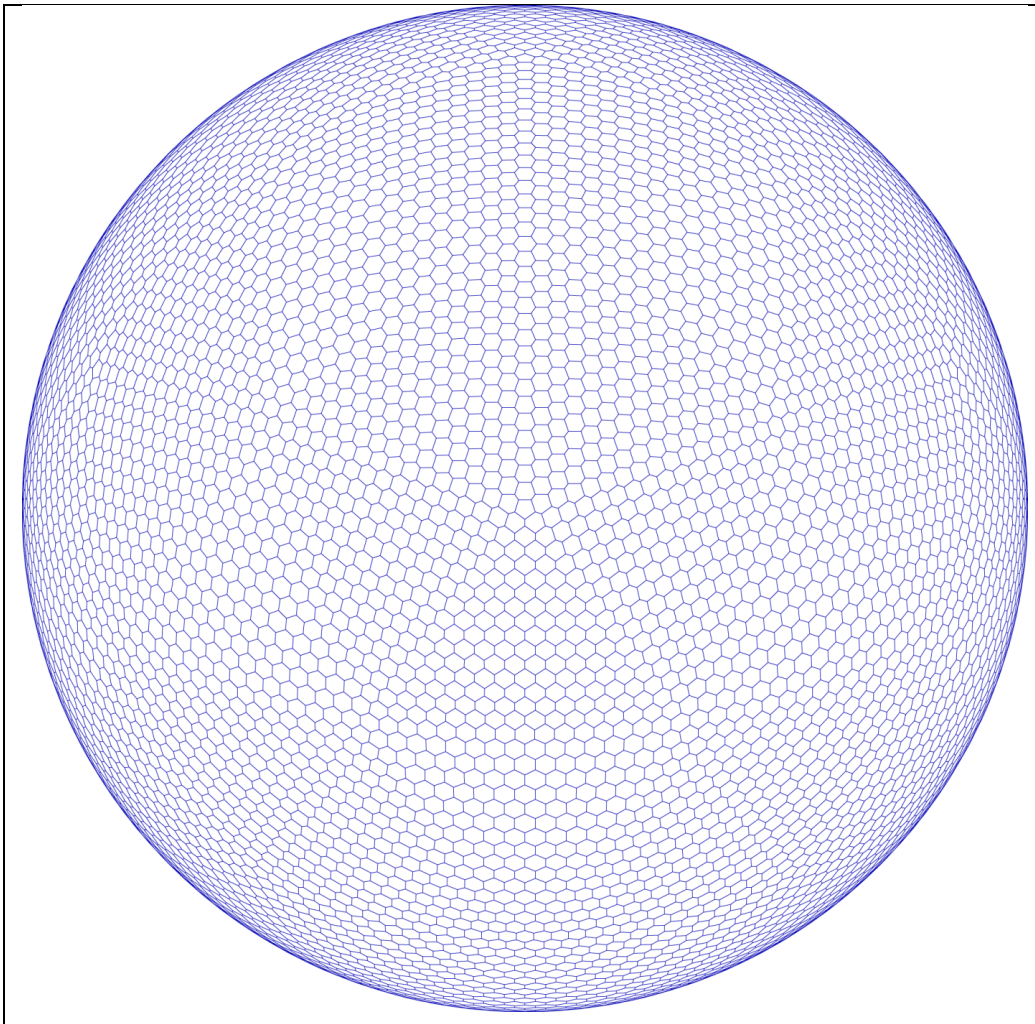

Figure 1: The icosahedral grid mesh composed of 10242 Voronoi cells (at grid refinement level 5).

Viewing Global Icosahedral Grids

There is a myriad of ways to display global model data, but we will only discuss a few here. One option is to use a plotting tool that renders an orthographic projection of the gridded data as a static image. This projection offers a natural view of a hemisphere from a given center point. Modelers typically create these types of plots when they need to debug a model or analyze specific grid points (see Figure 2).

Figure 2: The orthographic projection generated by a plotting tool. This projection gives a natural 2D representation of the sphere from a given center point.

A second option is to convert the global data into a static snapshot of the equidistant cylindrical projection, enabling the display of all data superimposed onto a single global map. This is perhaps the most common projection employed by FIM developers for displaying and comparing their model data [1] (see Figure 3).

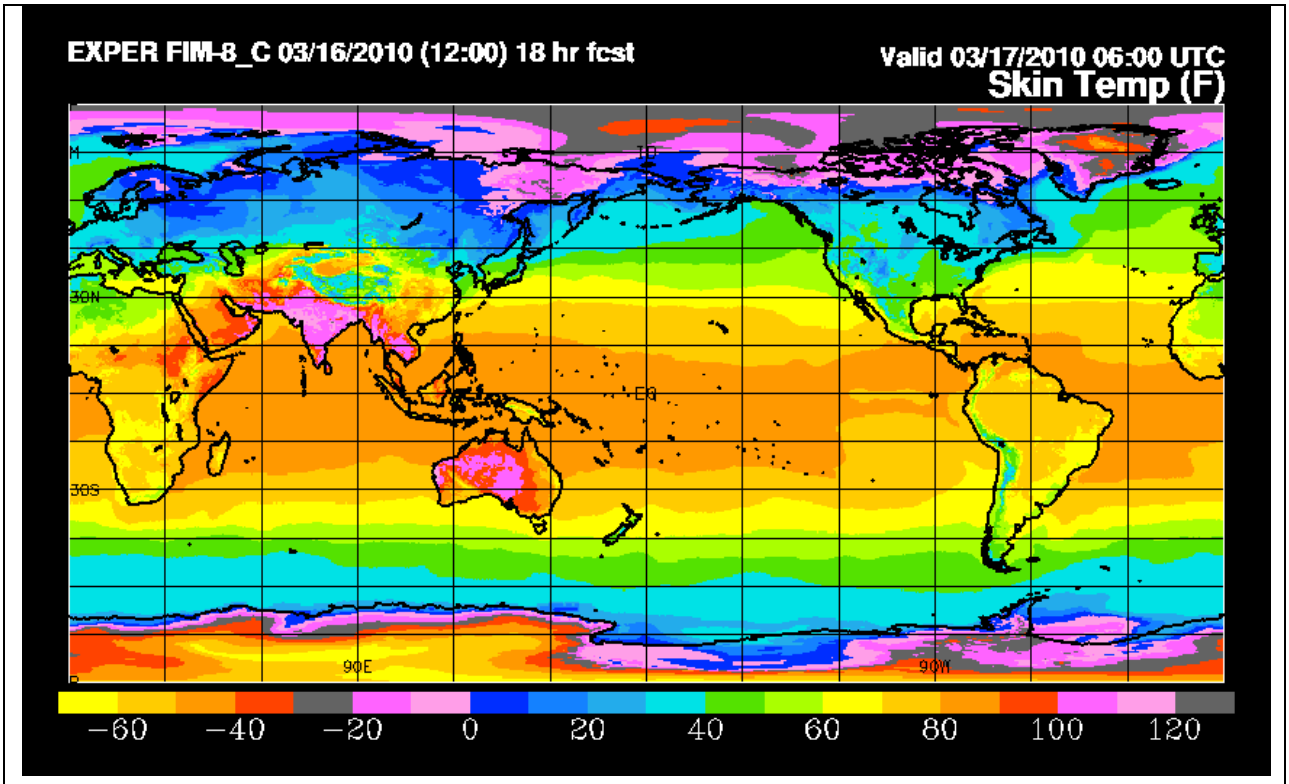

Figure 3: FIM Skin temperature mapped into a cylindrical equidistant projection with ½ degree resolution (with thanks to Brian Jamison). Distortion with this projection increases away from the Equator and is extreme in the polar regions.

A third option is to display the global data in Google Maps [3]. Google Maps uses a variant of the Mercator projection and does a good job of displaying data over much of the Earth, but with the same noticeable distortion as cylindrical equidistant projection (see Figure 4). Google Maps has the additional advantage of being interactive (i.e. dynamically controllable by the user).

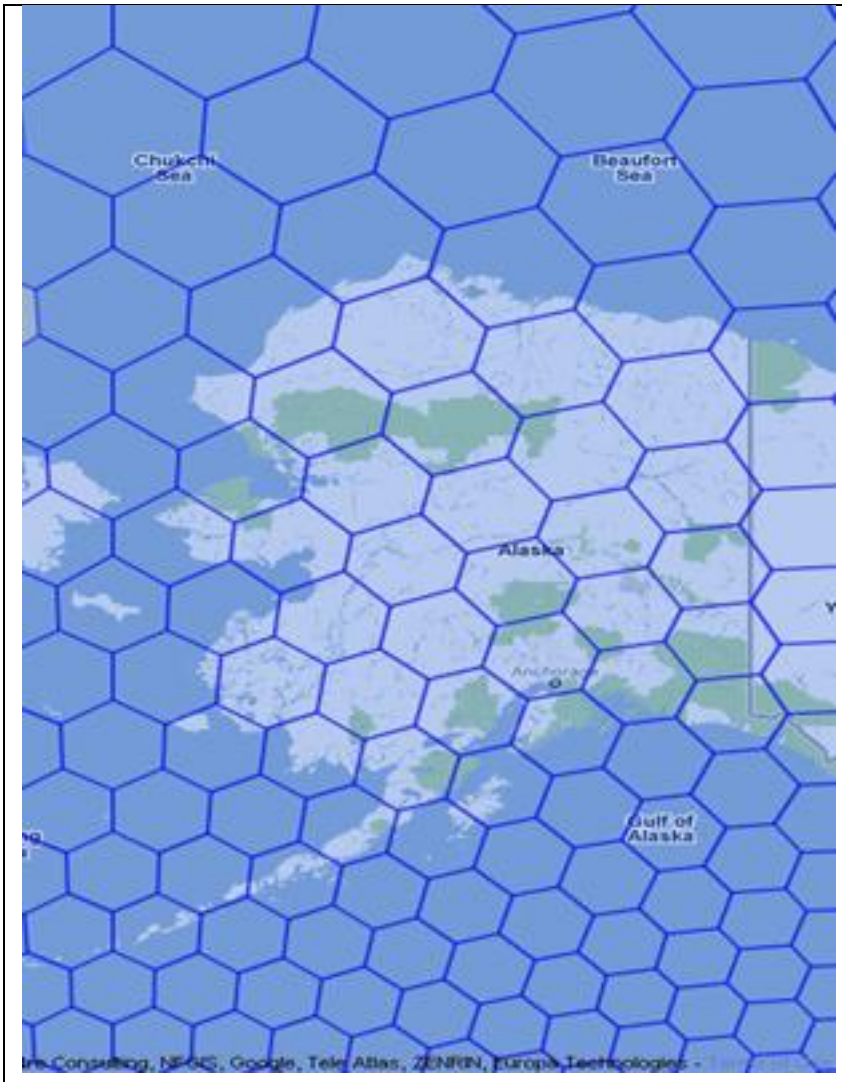

Figure 4: A portion of the FIM grid displayed in Google Maps, from roughly the 50th to the 75th parallel. Notice the shape distortion (of the otherwise area-equivalent icosahedral grid cells) increases towards the upper latitudes.

A fourth option for viewing global data is a virtual 3D globe like Google Earth [4]. The virtual globe allows users to view the earth from any center point in a quasi-orthographic projection. It does a good job of displaying data at any given latitude, while avoiding the distortion of Google Maps at polar areas. Also, like Google Maps, the display is interactive (see Figure 5).

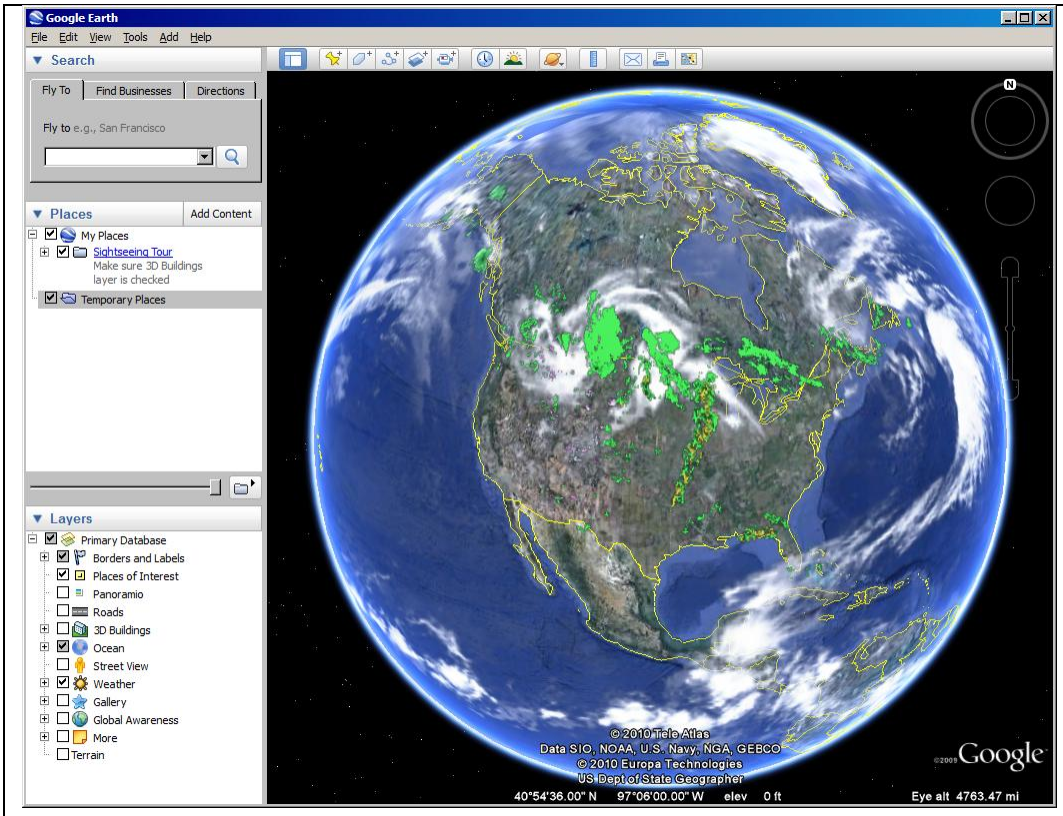

Figure 5: Google Earth employs a quasi-orthographic map projection.

Google Earth enables users to view their data as geospatial geometries or images that are superimposed as layers over satellite imagery. The software also allows for animation of the data, either over time (e.g. looping satellite imagery), or over space (e.g. animating vertical layers of forecast model data). Geometries such as icosahedral polygons and images are described and rendered in Google Earth via KML (Keyhole Markup Language), an XML-based language [5]. A code snippet of KML is included in Figure 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">

<LookAt>
        <longitude>-105.08</longitude>
        <latitude>40.56</latitude>
        <altitude>0</altitude>
        <range>50000</range>
        <tilt>0.00</tilt>
        <heading>0.0</heading>
</LookAt>
```

Figure 6: This KML code moves the Google Earth camera view to approximately 50000 meters above Fort Collins.

The two versions of Google Earth

The *Google Earth desktop* application is a stand-alone native client which runs on Windows, Mac OS/X and Linux. It loads and displays KML (or the compressed equivalent, KMZ files), and has a nice user interface with features such as bookmarks and rulers for measuring distances.

What the stand-alone version is to the desktop, the *Google Earth plug-in* is to the web browser. The Google Earth plug-in supports most of the features of the desktop version yet differs from the desktop in that it can only be viewed from within a web browser by being embedded into a web page. The plug-in supports a JavaScript API that enables developers to control the loading and display of data.

From a development perspective, there is no elegant way to create a customized application using the Google Earth desktop, let alone an application that is Internet accessible. Yet the plug-in allows for both customization and web integration. For this reason, it was decided that our research would focus on attempting to create a viewer using the Google Earth plug-in.

GIM Tool (Global Icosahedral Model Tool)

Our research and development effort culminated in the creation of two versions of GIM Tool: one version based on the Google Earth plug-in, and another based on Google Maps. Both versions allow users to select and subset various display fields (variables) from FIM, choose color palettes and map backgrounds, and control polygon edge visibility and fill opacity (how much of the background shows through the color-filled FIM polygons). Additionally, users can move the mouse over individual polygons to view details about each FIM cell.

Both versions retrieve data by invoking custom-created Representational State Transfer or RESTful web services [6], written in Java, which run in Apache Tomcat [7]. This Java code subsets the raw FIM data for the requested variable within the requested geographic region, and then builds a KML document to return to the calling (GIM Tool) client application. Rather than construct bitmap images to return in the KML, we decided to go a more flexible route and return KML polygons. KML polygons are advantageous because they can be encoded with metadata about FIM cells—metadata that can be accessed with a simple mouse click.

Experimentation determined that Google Earth can handle the rendering of tens of thousands, but not hundreds of thousands, of polygons and still perform adequately. The GIM Tool server was therefore optimized to return anywhere from two thousand to ten thousand polygons, at any given zoom level, which guaranteed acceptable performance.

To minimize the volume of FIM data being transferred to the clients, we down-scaled raw FIM 15 km resolution (G9) into lower resolutions, creating inter-grid mappings at resolutions of 30 km (G8), 60 km (G7), 120 km (G6), and 240 km (G5). As a point of reference, 240 km resolution is coarse enough to enable displaying FIM data over the entire globe with only a couple of thousand polygons. Testing revealed that the performance of this web service was good, with client requests for FIM data being fulfilled in as little as 36 milliseconds in the case of a single request, to as much as 74 milliseconds for 10 concurrent requests.

The Google Earth version of GIMTool employs the Google Web Toolkit (GWT) [8] and the Google Earth browser plug-in and only runs on Windows or Mac Safari browsers--the Google Earth browser plug-in does not currently support Linux. We chose to develop with the plug-in's JavaScript API by using GWT because it allowed us to write our source code in a language we prefer, Java, and then have the GWT compiler automatically convert this code into its JavaScript equivalent. The Google Earth version of GIM Tool supports true 3D views of FIM variables superimposed over a rotating globe, and is particularly useful for displaying grids in the upper latitudes and over the poles (see Figure 7).

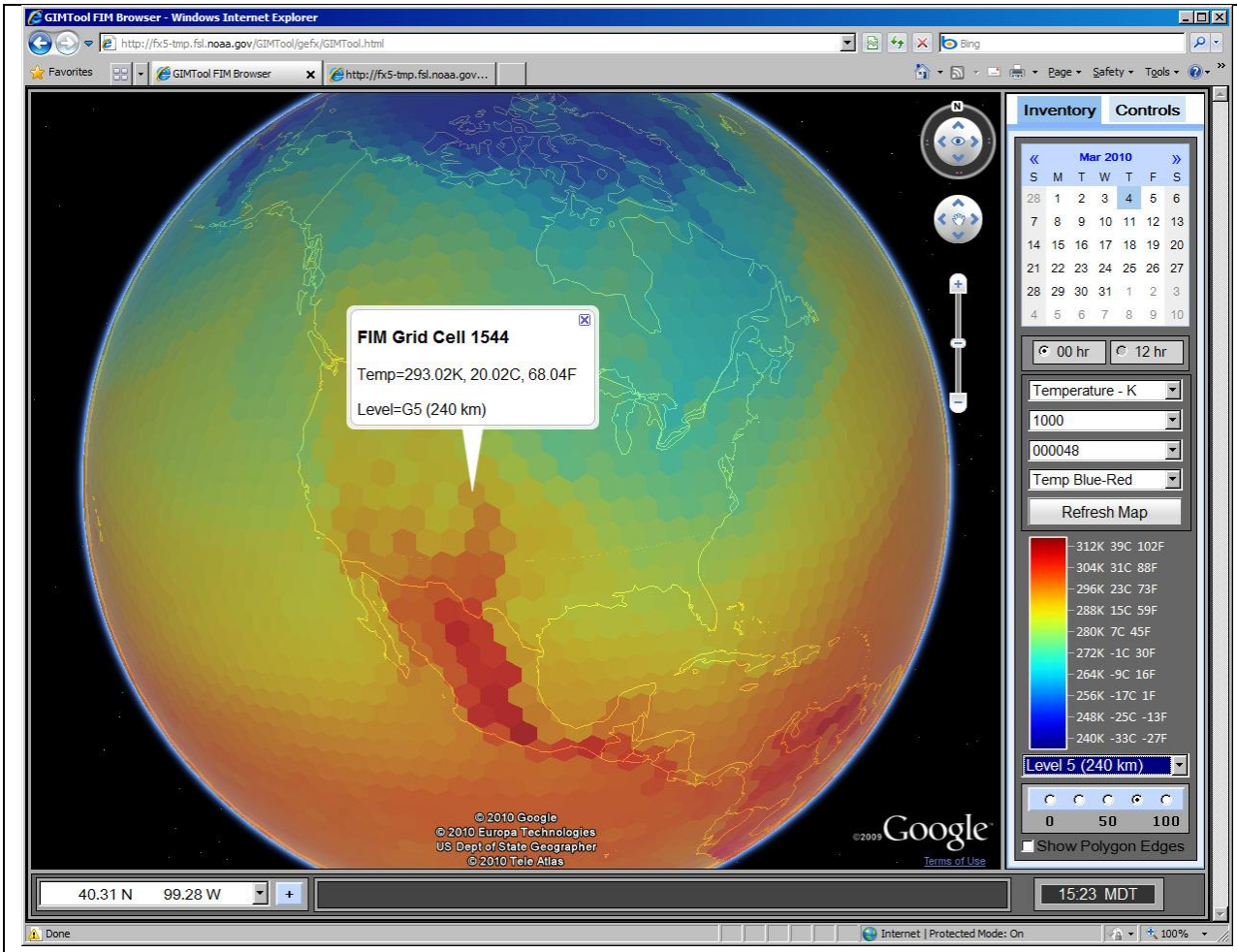

Figure 7: The Google Earth version of GIM Tool: a JavaScript based application that runs in a browser and uses the Google Earth plug-in to display FIM data.

The Google Maps version of GIM Tool was written with Flash Builder 4 and Google Maps. Flash Builder [9] was chosen because of its simple to use yet powerful graphical user interface builder, the fact that it creates SWF files compatible across all browsers and which looks the same on all browsers, and the fact that the underlying Actionscript language is object oriented and very similar to Java. The drawbacks of this Google Maps version, as compared to the Google Earth version, are that it displays data in the Google Maps Mercator-like projection (distorting the otherwise equal-area grid at the upper latitudes) and that it lacks

the “wow” factor of seeing a global dataset on a rotating globe. Advantages include faster load time, no Google Earth plug-in installation requirement, and support for virtually all browsers on all platforms (that support Flash). Moreover, when users “zoom in” to a regional and non-polar area, the display looks very similar to the Google Earth version’s display (see Figure 8).

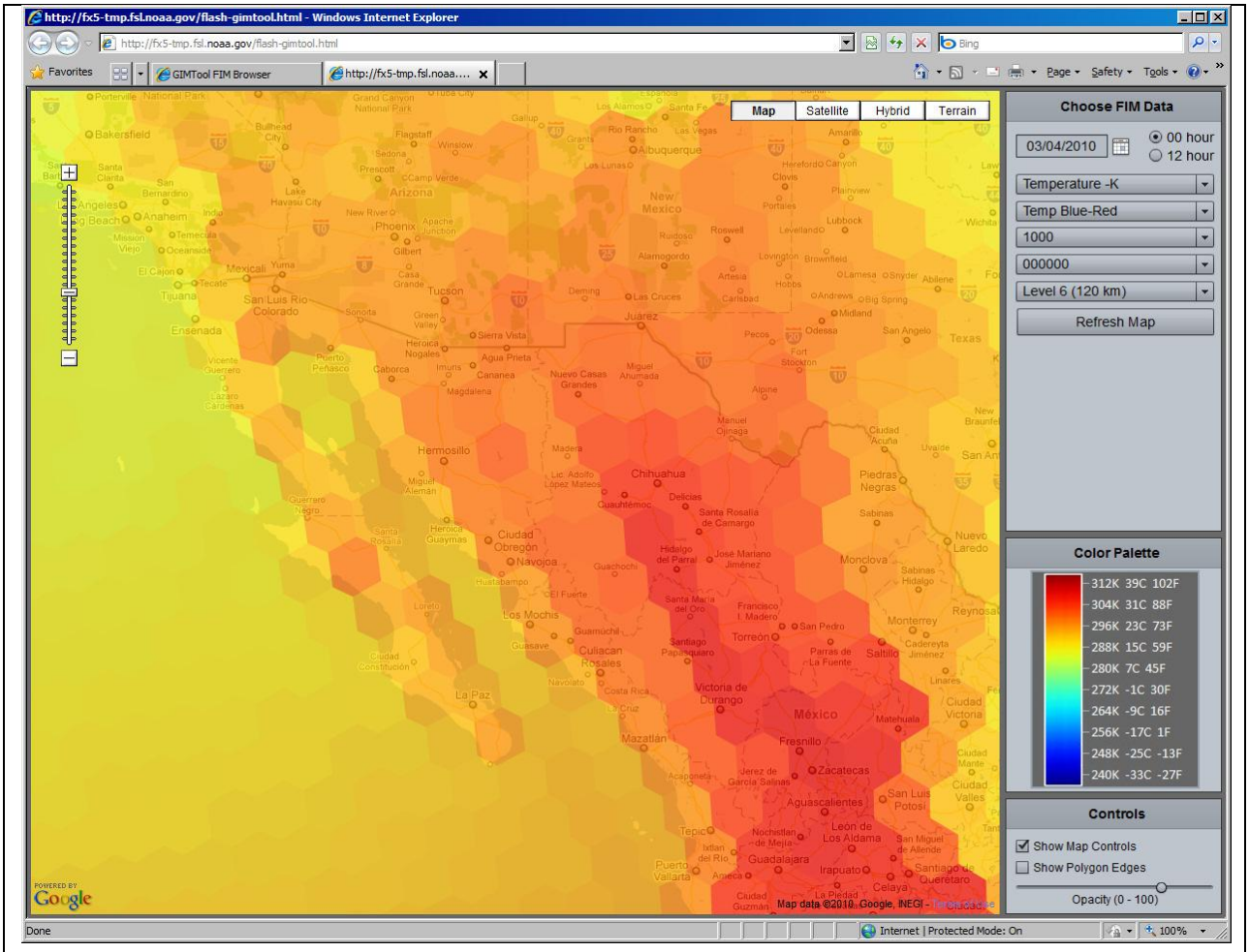

Figure 8: The Google Maps version of GIM Tool: a Flash based application that runs in a browser and uses Google Maps to display FIM data.

Unlike most viewers which typically generate static image files, GIM Tool is a dynamic tool with automatic progressive disclosure. For example, when a user drags (or pans) the map to the left or right, new data is automatically retrieved to redraw the new display region. If the user “zooms-in” (to a smaller region of the map), finer resolution grids are generated. When the user “zooms-out” (exposing a larger region of the map), coarser resolution grids are generated. In either case, the data the server stores and delivers are based on the grid resolution the global model is run at. For purposes of this research grid level 9 data was used.

The dynamic nature of GIM Tool both simplifies the user interface for the end user as well as makes for a more intuitive and immersive data visualization experience.

Future Work

The first versions of GIM Tool were warmly received by FIM researchers at GSD. Based on our meetings with the group, we plan to improve the tool in a number of ways:

- create a stand-alone version that doesn't require Tomcat
- add a dynamic palette editor
- support looping (animation)
- support additional FIM variables
- support overlaying other datasets such as vectors, contours, and shape files
- various user interface improvements
- support GSD's other global icosahedral model, NIM (Non-hydrostatic Icosahedral Model)

Acknowledgments

We'd like to acknowledge our esteemed colleagues: Mark Govett , Eric Hackathorn, Jacques Middlecoff, Vivian LeFebvre , and Doug Ohlhorst for their valuable contributions to the project.

References

[1] The finite-volume flow-following icosahedral model (http://fim.noaa.gov/FIMscp/)

[2] J.R. Baumgardner and P.O. Frederickson, “Icosahedral discretization of the two-sphere”, *SIAM J. Numer. Anal.*, **22**:1107-1115, Dec. 1985.

[3] Google Maps (http://maps.google.com/)

[4] Google Earth Official site (http://earth.google.com)

[5] KML Reference Documentation (http://code.google.com/apis/kml/documentation/kmlreference.html)

[6] Pautasso, Cesare; Zimmermann, Olaf; Leymann, Frank (2008-04), "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision", *17th International World Wide Web Conference (WWW2008)* (Beijing, China), http://www.jopera.org/docs/publications/2008/restws

[7] Apache Tomcat Server (http://tomcat.apache.org/)

[8] Google Web Toolkit Terms and Conditions (http://code.google.com/webtoolkit/terms.html)

[9] Official Adobe Flex site (http://www.adobe.com/products/flex)